On cooperative patrolling: optimal trajectories, complexity analysis, and approximation algorithms

Fabio Pasqualetti, Antonio Franchi, and Francesco Bullo

Abstract

The subject of this work is the patrolling of an environment with the aid of a team of autonomous agents. We consider both the design of open-loop trajectories with optimal properties, and of distributed control laws converging to optimal trajectories. As performance criteria, the *refresh time* and the *latency* are considered, i.e., respectively, time gap between any two visits of the same region, and the time necessary to inform every agent about an event occurred in the environment. We associate a graph with the environment, and we study separately the case of a chain, tree, and cyclic graph. For the case of chain graph, we first describe a minimum refresh time and latency team trajectory, and we propose a polynomial time algorithm for its computation. Then, we describe a distributed procedure that steers the robots toward an optimal trajectory. For the case of tree graph, a polynomial time algorithm is developed for the minimum refresh time problem, under the technical assumption of a constant number of robots involved in the patrolling task. Finally, we show that the design of a minimum refresh time trajectory for a cyclic graph is *NP-hard*, and we develop a constant factor approximation algorithm.

I. Introduction

The recent development in the autonomy and the capabilities of mobile robots greatly increases the number of applications suitable for a team of autonomous agents. Particular interest has been received by those tasks requiring continual execution, such as the monitoring of oil spills [1], the detection of forest fires [2], the track of border changes [3], and the patrol (surveillance) of an environment [4]. The surveillance of an area of interest requires the robots to continuously and repeatedly travel the environment, and the challenging problem consists in scheduling the robots trajectories so as to optimize a certain performance criteria. The reader familiar with network location, multiple traveling salesman, or graph exploration problems may observe a close connection with the patrolling problem we address, e.g., see [5], [6], [7]. It is worth noting, however, that these classical optimization problems do not capture the repetitive, and hence dynamic, aspect of the patrolling problem, nor the synchronization issues that arise when a timing among the visits of certain zones is required.

A precise formulation of the patrolling problem requires the characterization of the robots capabilities, of the environment to be patrolled, and of the performance criteria. In this work, we assume the robots to be identical and capable of sensing and communicating within a certain spatial range, and of moving according to a first order integrator dynamics with bounded speed. We represent the environment as a graph, in which the vertices correspond to physical and strategically important locations, and in which the edges denote the possibility of moving and communicating between locations. We assume that, when a robot is placed at each of the graph vertices, the union of the sensor footprints provides complete sensor coverage of the environment. Regarding the performance criteria of a patrolling trajectory, we consider (i) the time gap between any two visits of the same region, called *refresh time*, and (ii) the time needed to inform the team of robots about an event occurred in the environment, called *latency*. Loosely speaking, refresh time and latency reflect the effectiveness of a patrolling team in detecting events in the environment and in organizing remedial actions. For both the refresh time and latency optimization problem, we focus

This material is based upon work supported in part by NSF grant IIS-0904501 and CPS-1035917. The authors thank the reviewers for their thoughtful and constructive remarks.

Fabio Pasqualetti and Francesco Bullo are with the Center for Control, Dynamical Systems and Computation, University of California at Santa Barbara, {fabiopas,bullo}@engineering.ucsb.edu

Antonio Franchi is with the Department of Human Perception, Cognition and Action, Max Plank Institute for Biological Cybernetics, antonio.franchi@tuebingen.mpg.de

on the worst case analysis, even though the average refresh time and the average latency cases are also of interest. Notice that for the latency to be finite, the motion of the robots needs to be synchronized. For instance, if two robots are allowed to communicate only when they simultaneously occupy two adjacent vertices of the graph, then they need to visit those vertices at the same time in a finite latency trajectory.

The patrolling problem is receiving increasing attention because of its fundamental importance in many security applications, e.g., see [8], [9], [10], [11], [12]. Although many solutions have been proposed, the problem of designing minimum refresh time and latency team trajectories for a general environment is, to date, an open problem. Almost all traditional approaches rely on space decomposition, and traveling salesperson tour computation [13]. In [14] an empirical evaluation of existing patrolling heuristics is performed. In [15] two classes of strategies are presented, namely the cyclic- and the partition-based strategy. In the cyclic-based strategy, the robots compute a closed route through the viewpoints, and travel repeatedly such route at maximum speed. Clearly, in the case of a single robot, if the tour is the shortest possible, then the cyclic-based strategy performs optimally with respect to the refresh time and latency criteria. In the partition-based strategy, the viewpoints are partitioned into m subsets, being m cardinality of the team, and each robot is responsible for a different set of viewpoints. To be more precise, each robot computes a closed tour visiting the viewpoints it is responsible for, and it repeatedly moves along such tour at maximum speed. Still in [15], the two classes of strategies are compared, and it is qualitatively shown that cyclic-based strategies are to be preferred whenever the ratio of the longest to the shortest edge of the graph describing the environment is small, while, otherwise, partition-based policies exhibit better performance. In [4] and [2], an efficient and distributed solution to the perimeter patrolling problem for robots with zero communication range is proposed. By means of some graph partitioning and graph routing techniques, we extend the results along these directions, e.g., by considering the case of a nonzero communication range for the perimeter patrolling, and by characterizing optimal strategies for different environment topologies. An important variant of the patrolling problem is known as persistent surveillance, e.g., see [16]. Differently to our setup, a dynamically changing environment is considered for the persistent surveillance problem, and performance guarantees are offered only under a certain assumption on the rate of change of the regions to be visited.

It is worth mentioning that a different approach to the design of patrolling trajectories relies on the use of pebbles or chemical traces to mark visited regions, e.g., see [17], [18], [19], [20]. These techniques, although effective even without a global representation of the environment, and with severe communication constraints, do not explicitly deal with the optimality of the patrolling trajectories, and they represent therefore a complementary area of research with respect to this work.

The main contributions of this work are as follows. We introduce and mathematically formalize the concept of refresh time and latency of a team trajectory, and we formally state the patrolling optimization problem. We propose a procedure to build a graph (roadmap) to represent the topological structure of the area to be patrolled, and we study separately the case of a chain, tree, and cyclic (not acyclic) graph. We exhaustively discuss the case of a chain roadmap. First, we characterize a family of minimum refresh time and latency team trajectories, which can be computed by optimally partitioning the chain graph among the robots. Second, we derive a centralized polynomial time algorithm to compute an optimal partition, and, ultimately, to design an optimal team trajectory. Our partitioning procedure is based upon a bisection method, and it is also amenable to distributed implementation. Third, we develop a distributed procedure for the robots to converge and synchronize along an optimal trajectory, so as to minimize the refresh time and latency criteria. Fourth and finally, we test the robustness of our methods through a simulation study. When the roadmap has a tree or cyclic structure, we focus on the refresh time optimization problem, and we do not consider the latency optimization nor the design of distributed algorithms. For the case of a tree roadmap, we reduce the minimum refresh time patrolling problem to a known graph optimization problem. We show that the computational complexity of the minimum refresh time patrolling problem is polynomial in the number of vertices of the roadmap, and, under the assumption of a fixed and finite number of robots, we identify a polynomial time centralized algorithm to compute a minimum refresh time team trajectory. For the case of a cyclic roadmap, we show that the patrolling problem is an *NP-hard* optimization problem.

We propose two approximation algorithms, and we characterize their performance. The first approximate solution is extremely easy to compute, but its performance depends upon the ratio between the longest and the shortest edge in the graph representing the environment. The second approximation algorithm is based on a polynomial time path-covering procedure, and it allows us to compute a team trajectory whose refresh time is within a factor of 8 from the minimum refresh time for the given environment (cf. Fig. 13 for an example). To the best of our knowledge, this algorithm is the first constant factor approximation algorithm for the *NP-hard* minimum refresh time patrolling problem.

A preliminary version of this work appeared in [21]. With respect to the latter manuscripts, in this current work we introduce and solve the latency optimization problem, we perform a numerical study to analyze the robustness of our algorithmic procedures, and we improve the presentation of the results on the refresh time optimization problem.

The rest of the paper is organized as follows. The notation and the problem under consideration are in Section II, where we also show that the patrolling problem is, generally, computationally hard. Section III, IV, and V contain our results for the patrolling of a chain environment. We characterize a minimum refresh time and latency team trajectory, and we derive a centralized and a decentralized algorithm for its computation. In Section VI we perform a simulation study to show some robustness and reconfigurability properties of our distributed procedure. Section VII-A contains our results for the patrolling of a tree environment. We describe a minimum refresh time team trajectory on a tree roadmap, and we characterize the complexity of computing an optimal solution. Section VII-B deals with the general case of cyclic environment, and it contains our approximation procedures. Our conclusion and final remarks are in Section VIII.

II. Robotic model and preliminary concepts

A. Robots on roadmaps with sensor coverage and communication connectivity

We will be using the standard motion planning notation, and we refer the reader to [22] for a comprehensive treatment of the subject. We are given a team of $m > 0$ identical robots, capable of sensing, communicating, and moving in a connected environment. We make the following combined assumptions on the robot capabilities and on the environment to be patrolled.

Regarding sensing, we assume that the environment can be completely covered by simultaneously placing a robot at each of a set of $n > m$ *viewpoints* in the configuration space. In other words, if $m = n$ robots were available and placed at the n viewpoints, then the union of the sensors footprint of the robots would provide complete sensor coverage of the environment. We assume that each viewpoint is required for complete sensor coverage. Finally, we assume $n > m$ so that at least one robot needs to visit more viewpoints for the entire environment to be monitored over time.

Regarding communication, we associate an undirected graph G with the environment, whose vertices are the n viewpoints, and in which there is an edge between two vertices if two robots placed at those viewpoints are able to communicate to each other. We assume that G is connected. In what follows we design cooperative patrolling algorithms with sporadic communication, in the sense that two robots are required to communicate only when they occupy adjacent vertices. The occurrence of additional communication links can be easily incorporated into the algorithms and cannot decrease their performance.

Regarding motion, we assume that the robots are holonomic, i.e., they are modeled as first order integrators and move at most at unit speed. Additionally, we turn the graph G into a robotic roadmap [22] and a metric weighted graph as follows: to each pair of viewpoints that are neighbors in G, we associate a unique path connecting them. We select these paths so that the set of path lengths, adopted as edge weights, verify the triangle inequality. (For example, the shortest paths between viewpoints constitute a suitable choice.) We assume that each robot remains always on the roadmap.

In summary, the combined assumptions on the robot capabilities and on the environment are that: the vertices of G provide complete sensor coverage of the environment and each edge of G corresponds to both a communication edge and a motion path. Hence, we refer to G as a roadmap with sensor coverage and communication connectivity.

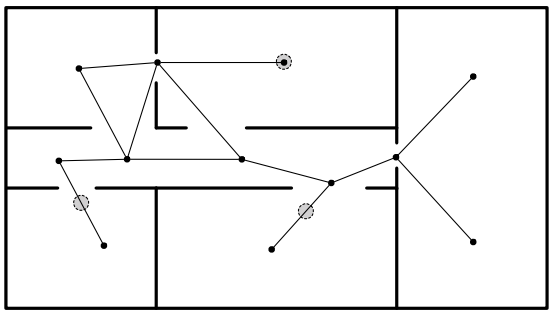


Fig. 1. A polygonal environment and its associated roadmap. The viewpoints are chosen in a way that the environment is completely covered by placing a robot at each of the 12 viewpoints. The edges of the roadmap denote the possibility for the 3 robots of both moving and communicating between pair of connected viewpoints. The weight of each edge corresponds to its length.

B. On suitable roadmaps and robots capabilities

The problem of constructing a roadmap from an environment is here discussed.

Example 1 (Roadmap computation with omnidirectional sensing and communication in known environments)**.** *Assume that the robots are holonomic vehicles moving at bounded speed, and equipped with an omnidirectional sensing device, and a line-of-sight communication device. If a map of the environment is available, then a valid roadmap is obtained by solving an Art Gallery Problem with Connectivity [23]. A solution to the Art Gallery Problem with Connectivity is a set of locations, called guards, with the following two properties: each point in the environment is visible by at least one guard and the visibility graph of the guards is connected. An example roadmap is given in Fig. 1. (A distributed sensor-based algorithm for the Art Gallery Problem with Connectivity is given in [24].)*

Example 2 (Roadmap computation for general robots in an unknown environment)**.** *If the environment is unknown, then an exploration algorithm should be used to obtain a representation of the environment, see for example [18], [25], [26]. While or after exploring the environment, a robot should select viewpoints to provide full sensor coverage. By construction, the existence of a path between two viewpoints is automatically guaranteed by the exploration algorithm. Moreover, if communication between two path-connected viewpoints (v_i, v_j) is not guaranteed, then the graph may be augmented with additional viewpoints along the path connecting v_i and v_j. The roadmap resulting from these steps features sensor coverage and communication connectivity.*

The following remarks are in order. First, a roadmap representing the environment is in general not unique, and the patrolling performance depends upon the particular choice. In this work, we do not address the problem of choosing the roadmap that yields optimal performance. Instead, we present efficient algorithms for the patrolling problem, which can also be used to compare different roadmaps on the basis of the corresponding patrolling performance. Second, for the implementation of our algorithms, a robot does not need to travel exactly along the roadmap. Indeed, a robot only needs to arrive sufficiently close to a viewpoint, or to be able to circle around it to provide sensor coverage. A related example is in Section VI, where the arrival delay of a robot can be interpreted as the uncertainty in the motion of the robots. Third, the global knowledge of the roadmap may not be required for the execution of a patrolling trajectory. Indeed, in general, each robot only visits a subpart of the roadmap. Fourth and finally, collisions are prevented by letting two robots exchange their roles every time they are about to collide. Indeed, since no robot is assigned to a specific viewpoint, if robot i is about to collide with robot j at time T, then, without affecting the performance of the team trajectory, collision is avoided by redefining the i-th and j-th trajectory as $\bar{x}_i(t) = x_j(t)$ and $\bar{x}_j(t) = x_i(t)$ for $t \geq T$; see the notion of order invariant trajectory below. Communication or sensing devices can be used to detect and prevent possible collisions.

C. Refresh time of team trajectories: definitions and a preliminary complexity result

For a team of m robots with specific capabilities, let $G = (V, E)$ be a robotic roadmap with sensor coverage and communication connectivity. A *team trajectory* X is an array of m continuous and piecewise-differentiable trajectories $x_1(t), \ldots, x_m(t)$ defined by the motion of the robots on the roadmap G, i.e., $x_i : [0, T_\mathrm{f}] \mapsto G$, for $i \in \{1, \ldots, m\}$, where $T_\mathrm{f} \in \mathbb{R}$ is a time horizon of interest, much larger than the time required by a single robot to visit all viewpoints. We say that a viewpoint $v \in V$ is visited at time t by robot i if $x_i(t) = v$. We define the *refresh time* of a team trajectory X, in short $\mathrm{RT}(X)$, as the longest time interval between any two consecutive visits of any viewpoint, i.e.,

$$\mathrm{RT}(X) = \max_{v \in V} \max_{(t_1, t_2) \in \Omega(v, X)} t_2 - t_1$$

where $\Omega(v, X) = \{(t_1, t_2) \in [0, T_\mathrm{f}]^2, t_1 \leq t_2, \mid x_i(t) \neq v, \forall i \in \{1, \ldots, m\}, t_1 < t < t_2\}$.

Remark 1 (Existence of a minimum)**.** *We claim that there exists a team trajectory with minimum refresh time and prove it as follows. Without loss of generality, we restrict our attention to team trajectories in which each robot moves at maximum speed along the edges and stops for certain durations, possibly equal to zero, at the viewpoints. Thus, a team trajectory can be described as a tuple of the form* $(S, \Delta) = \{(S_1, \Delta_1), \ldots, (S_m, \Delta_m)\}$*, where* S_i *contains the sequence of viewpoints visited by robot* i*, and* Δ_i *contains the waiting times at the visited vertices. Notice that the time horizon* T_f *is finite, the length of each edge is positive, the number of vertices is finite, and the speed of the robots is bounded. It follows that the length of each sequence* S_i *is finite, and, therefore, each* S_i *takes value in a finite set. Now, for each possible sequence of visited vertices, the refresh time is a continuous function of only the waiting times, and each waiting time lies in the compact interval* $[0, T_\mathrm{f}]$*. Because any continuous function defined over a compact region admits a point of minimum value, we conclude that there exists a team trajectory with minimum refresh time.*

Problem 1 (Team refresh time)**.** *Given a roadmap and a team of robots, find a minimum refresh time team trajectory.*

In Section IV we present a different optimization problem, which deals with the possibility for a robot to communicate, possibly with multiple hops, with every other robot in the team. We now conclude this section with our first result on the computational complexity of the Team refresh time problem. For a detailed discussion of the main aspects of the computational complexity theory, we refer the interested reader to [27]. Recall that an optimization problem is said to be *NP-hard* if it is, informally, as hard as the hardest problem in the class *NP*, for which no polynomial time algorithm is known to compute an optimal solution.

Theorem II.1 (Computational complexity)**.** *The* Team refresh time *problem is NP-hard.*

Proof: This statement can be shown by reduction from the Traveling Salesman problem [27]. In fact, if $m = 1$, since the speed of the robots is bounded, then a minimum refresh time trajectory consists of moving the robot at maximum speed along a shortest closed tour visiting the viewpoints. The problem of finding a shortest tour through a set of points in the plane, also known as Traveling salesman problem, is an *NP-hard* problem [27]. Hence, by restriction, the Team refresh time problem is also *NP-hard*. ■

Following Theorem II.1, the minimum refresh time optimization problem is generally computationally hard. In this work, we first identify two roadmap structures for which there exists an efficient solution to the Team refresh time problem, and then we describe two approximation algorithms to deal with the general case.

III. Minimum refresh time team trajectory on a chain roadmap

We characterize in this section an optimal refresh time team trajectory when the roadmap associated with the environment has a chain structure.

A. Open loop team trajectory characterization

Let N_i denote the neighbor set of the vertex i, and let $|N_i|$ denote the degree of i, i.e., the cardinality of the set N_i. A chain roadmap is an undirected, connected, and acyclic roadmap, in which every vertex has degree two, except for two vertices which have degree one. Without losing generality, we assume that the n vertices are ordered in a way that $N_1 = \{2\}$, $N_n = \{n-1\}$, and $N_i = \{i-1, i+1\}$ for each $i \in \{2, \dots, n-1\}$. We define a relative order of the robots according to their position on the roadmap. A team trajectory is *order invariant* if the order of the robots does not change with time, i.e., if $x_i(t) \leq x_{i+1}(t)$ for each $i \in \{1, \dots, m-1\}$ and for every instant $t \in [0, T_\mathrm{f}]$, where $x_i(t)$ denotes the distance at time t on the roadmap from the first vertex of the chain to the position of the i-th robot.

Proposition III.1 (Order invariant team trajectory)**.** *Let X be a team trajectory. There exists an order invariant team trajectory $\bar{X}$ such that RT$(X) =$ RT$(\bar{X})$.*

Proof: Let X be a team trajectory, and consider the permutation matrix $P(t)$, that keeps track of the order of the robots at time t, i.e., such that the (i,j)-th entry of $P(t)$ is 1 if, at time t, the i-th robot occupies the j-th position in the chain of robots, and it is 0 otherwise. Since X is continuous, anytime the function $P(t)$ is discontinuous, the positions of the robots directly involved in the permutation overlap. Therefore, the order invariant team trajectory $\bar{X} = P^{-1}(t)X(t)$ is a feasible team trajectory, and it holds $\mathrm{RT}(\bar{X}) = \mathrm{RT}(X)$. ∎

Let $V_i \subseteq V$ be the set of viewpoints visited over time by the agent i with the trajectory x_i, and let the *image* of the team trajectory X be the set $\{V_1, \dots, V_m\}$. Notice that different team trajectories may have the same image. Throughout the paper, let $l_i = \min_{v \in V_i} v$, $r_i = \max_{v \in V_i} v$, and $d_i = r_i - l_i$. Finally, let $\mathrm{RT}^* = \min_X \mathrm{RT}(X)$. A team trajectory is *non-overlapping* if $V_i \cap V_j = \emptyset$ for all $i \neq j$.

Proposition III.2 (Non-overlapping team trajectory)**.** *Given a chain roadmap, there exists an order invariant and non-overlapping team trajectory with refresh time RT*.*

Proof: Let X^* be a minimum refresh time team trajectory, and let X be the order invariant team trajectory obtained from X^* as in Proposition III.1. Clearly $\mathrm{RT}(X) = \mathrm{RT}^*$. Let $\{V_1, \dots, V_m\}$ be the image of X, and note that $V = \cup_{i=1}^m V_i$. Consider the partition of V defined as

$$\begin{aligned} \bar{V}_1 &= V_1, \\ \bar{V}_i &= V_i \setminus \cup_{j=1}^{i-1} V_j, \quad i \in \{2, \dots, m\}. \end{aligned}$$

For every nonempty $\bar{V}_i$, let $\bar{l}_i = \min_{v \in \bar{V}_i} v$, $\bar{r}_i = \max_{v \in \bar{V}_i} v$, and $\bar{d}_i = \bar{r}_i - \bar{l}_i$. Note that, by construction, the viewpoint $\bar{l}_i$ is visited by the robot i and, possibly, by the robots $j > i$. Also, because X is order invariant, we have $x_i(t) \leq x_j(t)$. It follows that $\mathrm{RT}(X) \geq 2 \max_i \bar{d}_i$. Consider now the team trajectory $\bar{X}$ with image $\{\bar{V}_1, \dots, \bar{V}_m\}$, and assume that the robots sweep periodically at maximum speed their segment. Then $\mathrm{RT}(\bar{X}) = 2 \max_i \bar{d}_i$, so that $\bar{X}$ is an order invariant and non-overlapping team trajectory with minimum refresh time. ∎

Given a chain graph on the viewpoints V, let $\Pi_m = \{\pi_1, \dots, \pi_m\}$ be an *m-partition* of V, i.e., $\pi_1, \dots, \pi_m$ is a collection of subsets of V such that $\pi_i \cap \pi_j = \emptyset$ whenever $i \neq j$, and $V = \bigcup_{i=1}^m \pi_i$. Additionally, let the dimension of the partition Π_m equal the longest distance between any two viewpoints in the same cluster, i.e., $\dim(\Pi_m) = \max_{i \in \{1,\dots,m\}} (\max_{v \in \pi_i} v - \min_{v \in \pi_i} v)$, where $\max_{v \in \pi_i} v - \min_{v \in \pi_i} v = 0$ if $\pi_i = \emptyset$. Following Proposition III.2, there exists a minimum refresh time team trajectory whose image coincide with an m-partition of V. We now show that the minimum refresh time equals twice the dimension of an optimal m-partition.

Theorem III.3 (Minimum refresh time)**.** *Let G be a chain roadmap, and let m be the number of robots. Then RT* $= 2 \min_{\Pi_m} \dim(\Pi_m)$.*

Proof: As a consequence of Propositions III.1 and III.2, there exists a minimum refresh time team trajectory whose image coincides with an m-partition Π_m. Since each robot is assigned a different cluster,

Trajectory 1: *Minimum refresh time trajectory on a chain roadmap (i-th robot)*

Input : $l_i := \min_{v \in V_i} v$, $r_i := \max_{v \in V_i} v$, $d_i := r_i - l_i$;
Require : an optimal partition of the chain graph;

1: $x_i(t) := l_i$ for $t := 0, 2d_i, 4d_i, \dots$;
2: $x_i(t) := r_i$ for $t := d_i, 3d_i, 5d_i, \dots$;

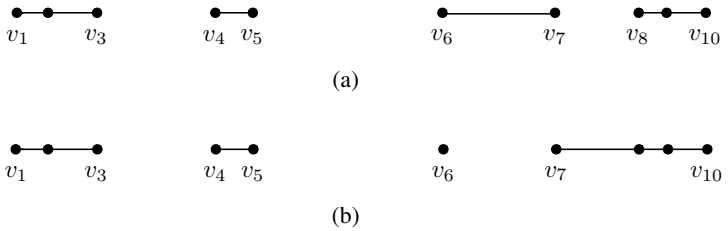


Fig. 2. A 4-partition with minimum (maximum) dimension is in Fig. 2(a). The dimension of this partition is $v_7 - v_6$. A 4-partition with minimum (average) dimension is obtained by removing 3 longest edges and it is reported in Fig. 2(b). The dimension of this partition is $v_{10} - v_7 > v_7 - v_6$.

and the speed of the robots is bounded by 1, we have $\text{RT}^* \geq 2\text{dim}(\Pi_m)$. Consider a team trajectory X in which each robot continuously sweeps at maximum speed the cluster it is assigned to. Clearly, $\text{RT}(X) = \text{RT}^* = 2\text{dim}(\Pi_m)$. ■

We have shown that a minimum refresh time trajectory consists of letting the robot sweep at maximum speed a part of the chain graph. Such a trajectory is more formally described for the i-th robot in Trajectory 1, where we only characterize the instants of time at which robot i changes its velocity vector, and we assume that it moves at maximum speed otherwise.

Remark 2 (Average partition)**.** *By removing the longest edges in the chain the average length of the clusters is minimized. In general, such partition does not minimize the dimension of the m-partition, and hence it is not optimal in our sense. An example is in Fig. 2.*

B. Optimal m-partition centralized computation

In the remaining part of the section we describe an algorithm to compute an optimal m-partition. For a set of viewpoints V, we call *left-induced* partition of length $\rho \in \mathbb{R}_{\geq 0}$ the partition $\Pi^\rho = \{\pi_i\}$ defined recursively as (cf. Fig. 3(a))

$$\pi_i = \{v \in V \mid a_i \leq v \leq a_i + \rho\}, \quad i = 1, \dots, \tag{1}$$

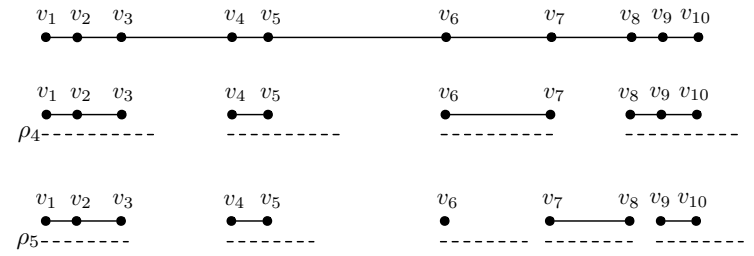
where

$$\begin{aligned} a_1 &= v_1, \\ a_j &= \min\{v \in V \mid v > a_{j-1} + \rho\}, \quad j = 2, \dots . \end{aligned}$$

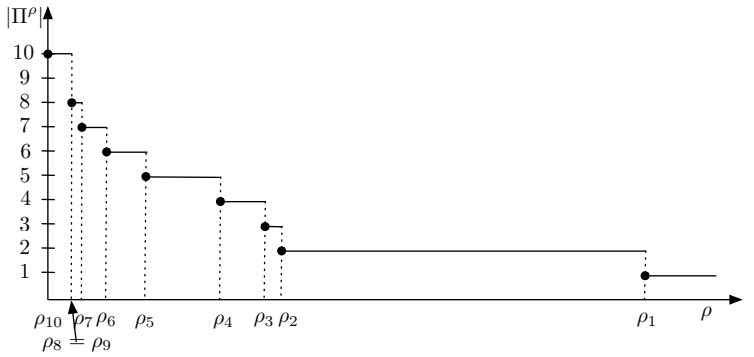
The cardinality $|\Pi^\rho|$ corresponds to the integer j such that $\{v \in V \mid v > a_j + \rho\} = \emptyset$. Observe that the function $\rho \mapsto |\Pi^\rho|$ is monotone, non-increasing, and right-continuous (cf. Fig. 3(b)). Let $\{\rho_1, \dots, \rho_{n-1}\} \in \mathbb{R}^{n-1}_{\geq 0}$ be the discontinuity points of the function $\rho \mapsto |\Pi^\rho|$, then, for $k \in \{1, \dots, n-1\}$,

$$\begin{aligned} |\Pi^\rho| &\leq k, \quad \text{if } \rho \geq \rho_k, \\ |\Pi^\rho| &> k, \quad \text{if } \rho < \rho_k. \end{aligned} \tag{2}$$

Note that two or more discontinuity points of $|\Pi^\rho|$ may coincide, so that the function $|\Pi^\rho|$ may not assume all the values in the set $\{1, \dots, n\}$, e.g., in Fig. 3(b) the value $|\Pi^\rho|$ is never equal to 9.



(a)



(b)

Fig. 3. In 3(a) the left-induced partition of length ρ_4 and ρ_5, with $\rho_5 < \rho_4$, for the chain roadmap with vertices $\{v_1, \dots, v_{10}\}$. The cardinalities are $|\Pi^{\rho_4}| = 4$ and $|\Pi^{\rho_5}| = 5$, respectively. In 3(b) the cardinality $|\Pi^\rho|$ is plotted as a function of the length ρ. Notice that, because $v_2 - v_1 = v_{10} - v_9$, the function $|\Pi^\rho|$ does not assume the value 9.

Algorithm 2: Optimal left-induced m-partition

Input : Viewpoints $v_1, \dots, v_n$, Number of robots $m \leq n$, Tolerance $0 < \varepsilon < \frac{v_n}{m}$;
Set : $a := 0$, $b := \frac{2v_n}{m}$, $\rho := \frac{(a+b)}{2}$;

```
while (b − a) > 2ε do
    Π^ρ := left-induced({v_1, ..., v_n}, ρ);
    if |Π^ρ| > m then
        a := ρ, ρ := (a+b)/2;
    else
        Π* := Π^ρ, b := ρ, ρ := (a+b)/2;
return Π*
```

Theorem III.4 (Optimal m-partition)**.** *Let G be a chain roadmap. Let Π_m be an m-partition of G, and let Π^ρ be the left-induced partition of length ρ of G. Then*

$$\min_{\Pi_m} \dim(\Pi_m) = \min\{\rho \in \mathbb{R}_{\geq 0} \mid |\Pi^\rho| \leq m\}.$$

Proof: Let Π_m be an m-partition, and let $\Pi^\rho = \{\pi_1^\rho, \dots, \pi_k^\rho\}$ be the left induced partition of length ρ of a chain roadmap G. Let $\rho^* = \min_{\Pi_m} \dim(\Pi_m)$. We want to show that ρ^* is one of the discontinuity points of the function $|\Pi^\rho|$, i.e., that ρ^* verifies the conditions (2).

By contradiction, if $\rho < \rho^*$ and $|\Pi^\rho| \leq m$, then an m-partition with dimension smaller than the optimal would exists. Therefore we have $|\Pi^\rho| > m$ if $\rho < \rho^*$.

Suppose now that $\rho \geq \rho^*$, and let $\Pi_m^* = \{\pi_1^*, \dots, \pi_m^*\}$ be an m-partition with minimum dimension. Notice that $|\pi_1^\rho| \geq |\pi_1^*|$, because the cluster π_1^ρ contains all the viewpoints within distance ρ from v_1, and hence also within distance ρ^*. It follows that $\max \pi_1^\rho \geq \max \pi_1^*$, and also that $\min \pi_2^\rho \geq \min \pi_2^*$. By repeating the same procedure to the remaining clusters, we obtain that $\max \pi_m^\rho \geq \max \pi_m^*$, so that, if $|\Pi^*| = m$ and $\rho \geq \rho^*$, then $|\Pi^\rho| \leq m$. ■

Following Theorem III.4, an optimal left-induced partition of cardinality (at most) m is also an optimal m-partition. Notice that for the computation of an optimal left-induced partition only the lengths ρ corresponding to the discontinuity points of Π^ρ need to be considered. Since each discontinuity point coincides with the distance between a pair of vertices, only $\frac{n(n-1)}{2}$ values need to be tested. Therefore, an optimal left-induced partition can be computed with complexity $O(n^2)$. In what follows we describe an ε-approximation algorithm with linear complexity for any $\varepsilon \in \mathbb{R}_{>0}$. Notice that ε-approximation algorithms with linear complexity are often more convenient for a practical implementation than exact algorithms with higher complexity [7].

We now present our algorithm for the computation of an optimal m-partition. Since the function $\rho \mapsto |\Pi^\rho|$ is monotone and continuous, a bisection method is effective for finding its discontinuity points, and, therefore, for determining the shortest length of a left-induced partition of cardinality m. A bisection based procedure to compute an optimal left-induced partition is in Algorithm 2, where the function *left-induced($\{v_1, \ldots, v_n\}$,ρ)* returns the left-induced partition defined in equation (1). We next characterize the convergence properties of Algorithm 2.

Lemma III.5 (Convergence of Algorithm 2)**.** *Let G be a chain roadmap, and let Π_m denote an m-partition of G. Let $\rho^* = \min_{\Pi_m} dim(\Pi_m)$. Algorithm 2 with tolerance ε returns a left-induced partition of dimension at most $\rho^* + \varepsilon$ and cardinality at most m. Moreover, the time complexity of Algorithm 2 is $O(n \log(\varepsilon^{-1}))$.*

Proof: Algorithm 2 searches for the minimum length ρ^* that generates a left-induced partition of cardinality at most m. Because of Theorem III.4, the length ρ^* coincides with one of the discontinuity points of the function $|\Pi^\rho|$, and it holds $\rho^* \in (0, 2v_n/m)$. Indeed, $\rho^* > 0$ because $m < n$, and $\rho^* < 2v_n/m$, because $(2v_n/m)m > v_n$. Recall from (2) that $|\Pi^\rho| > m$ for every $\rho < \rho^*$, and that the function $\rho \mapsto |\Pi^\rho|$ is monotone. Note that the interval $[a, b]$, as updated in Algorithm 2, contains the value ρ^* at every iteration. The length of the interval $[a, b]$ is divided by 2 at each iteration, so that, after $\log_2(\frac{2v_n}{\varepsilon m})$, the value ρ^* is computed with precision ε. Since the computation of $|\Pi^\rho|$ can be performed in $O(n)$ operations, the time complexity of Algorithm 2 is $O(n \log(\varepsilon^{-1}))$. ■

As a consequence of Proposition III.2 and Theorem III.3, in what follows we only consider team trajectories whose image coincide with an m-partition. Therefore, for ease of notation, we use the set $\{V_1, \ldots, V_m\}$ to denote both the image set of a team trajectory and an m-partition of the chain graph. We conclude this section with a summary of the presented results.

Theorem III.6 (Patrolling a chain graph at minimum refresh time)**.** *Let G be a chain graph with n viewpoints and let $m \leq n$ be the number of robots. Let $\mathcal{V}$ be an optimal m-partition of G computed by means of Algorithm 2 with tolerance ε. Let $d_{\max}$ be the dimension of $\mathcal{V}$. A team trajectory with image $\mathcal{V}$, and minimum refresh time $2d_{\max}$ is computed as in Trajectory 1. Moreover, the time complexity of designing such trajectory is $O(n \log(\varepsilon^{-1}))$.*

IV. Minimum refresh time and latency team trajectory on a chain roadmap

The previous section considers the problem of designing team trajectories with optimal refresh time on a chain graph. In a patrolling mission it may be important for the robots to communicate with each other in order to gather information about the status of the entire environment. For instance, messages could be sent by a unit to ask for reinforcement, or to spread an alarm. Informally, we call *latency* of a team trajectory X, in short $\mathrm{LT}(X)$, the shortest time interval necessary for a message generated by any robot to reach all the other robots. In other words, given our communication model, the latency of a team trajectory is a measure of how fast a message spreads to all robots. In this section we describe team trajectories with minimum refresh time and latency.

We now give a more formal definition of $\mathrm{LT}(X)$. Recall that, by assumption, two robots are allowed to communicate when they lie on two adjacent viewpoints. In a chain roadmap, for a message to reach every robot in the chain, every pair of adjacent robots needs to communicate. For $i \in \{2, \ldots, m\}$, let Φ_i denote

the union of the set of times at which the robots $i-1$ and i communicate and $\{0\}$. The *up-latency* of X, in short $\mathrm{LT}_{\mathrm{up}}(X)$, is the longest time interval between any two consecutive communications between the robots $1, 2$ and $m-1, m$. Precisely,

$$\mathrm{LT}_{\mathrm{up}}(X) = \max_{t_2 \in \Phi_2} \min_{t_m \in \bar{\Phi}_m(t_2)} t_m - t_2,$$

where $\bar{\Phi}_m(t_2) = \{t_m \in \Phi_m \mid \exists\, t_3 \in \Phi_3, \dots, t_{m-1} \in \Phi_{m-1}, t_2 \le t_3 \le \cdots \le t_m\} \cup \{T_{\mathrm{f}}\}$. Analogously, we call *down-latency* the quantity

$$\mathrm{LT}_{\mathrm{down}}(X) = \max_{t_m \in \Phi_m} \min_{t_2 \in \bar{\Phi}_2(t_m)} t_2 - t_m,$$

where $\bar{\Phi}_2(t_m) = \{t_2 \in \Phi_2 \mid \exists\, t_3 \in \Phi_3, \dots, t_{m-1} \in \Phi_{m-1}, t_2 \ge t_3 \ge \cdots \ge t_m\} \cup \{T_{\mathrm{f}}\}$. Finally, we define the latency of a team trajectory as

$$\mathrm{LT}(X) = \max\{\mathrm{LT}_{\mathrm{up}}(X), \mathrm{LT}_{\mathrm{down}}(X)\}.$$

Notice that our definitions of latency hold for $m \ge 2$, and that, if $m = 2$, then we have $\mathrm{LT}_{\mathrm{up}}(X) = \mathrm{LT}_{\mathrm{down}}(X) = \mathrm{LT}(X) = 0$ for every team trajectory X. We envision that the up- and down-latency performance criteria should be adopted when it is of interest to report the status of the monitored area to a base station located at one end of the chain environment. The latency minimization problem is more appropriate for fully distributed scenarios. In this section we design synchronized team trajectories with the following two features. First, since a minimum refresh time trajectory is determined by an optimal partition of the chain graph, we aim at finding team trajectories associated with the same optimal partition.[1] Second, we design synchronized team trajectories with minimum up-latency (resp. down-latency) or latency.

A. Lower bound and optimal team trajectory for up-latency

We start by showing a lower bound for $\mathrm{LT}_{\mathrm{up}}(X)$ and $\mathrm{LT}_{\mathrm{down}}(X)$. Recall that, for a partition $\{V_1, \dots, V_m\}$, we have $l_i = \min_{v \in V_i} v$, $r_i = \max_{v \in V_i} v$, $d_i = r_i - l_i$, and $d_{\max} = \max_{i \in \{1,\dots,m\}} d_i$.

Lemma IV.1 (Up-latency lower bound)**.** *Let G be a chain roadmap, and let $\{V_1, \dots, V_m\}$ be an m-partition of G. The latency of a team trajectory with image $\{V_1, \dots, V_m\}$ is lower bounded by $\sum_{i=2}^{m-1} d_i$.*

Proof: The proposition follows from the fact that the robots speed is bounded by 1, and that the robots need to travel their segment to communicate with the neighboring robots. ■

For the up-latency of a team trajectory to equal the lower bound in Lemma IV.1, each robot i needs to transfer a message from robot $i-1$ to robot $i+1$ in time d_i. In order to do so, each robot i needs to communicate with its neighbor $i+1$ as soon as $x_i(t) = r_i$.

Theorem IV.2 (Patrolling a chain graph at minimum refresh time and minimum up-latency)**.** *Let G be a chain graph with n viewpoints and let $m \le n$ be the number of robots. Let $\mathcal{V}$ be an optimal m-partition of G computed by means of Algorithm 2 with tolerance ε. Let $d_{\max}$ be the dimension of $\mathcal{V}$, and let d_i be the length of the i-th cluster. A team trajectory with image $\mathcal{V}$, minimum refresh time $2d_{\max}$, and minimum up-latency $\sum_{j=2}^{m-1} d_j$ is computed as in Trajectory 3. Moreover, the time complexity of designing such trajectory is $O(n \log(\varepsilon^{-1}))$.*

Proof: The theorem follows by observing that the trajectory is $2d_{\max}$-periodic, and that no robot i waits at r_i to communicate with the neighboring robot $i+1$. The up-latency equals the lower bound in Lemma IV.1, and it is therefore minimum. Regarding the computational complexity, notice that it is determined by the computation of the optimal m-partition $\mathcal{V}$, and hence, by Lemma III.5, it equals $O(n \log(\varepsilon^{-1}))$. ■

An example of a team trajectory with minimum refresh time and minimum up-latency is in Fig. 4. Finally, observe that the minimization of the down-latency can be achieved in an analogous way.

Trajectory 3: *Minimum base-latency team trajectory (i-th robot)*

Input : $l_i := \min_{v \in V_i} v$, $r_i := \max_{v \in V_i} v$, $d_i := r_i - l_i$, $d_{\max} := \max_j r_j - l_j$;
Require : optimal partition of the chain graph;
Set : $t_0 := -\sum_{j=1}^{m-1} d_j$, $k \in \mathbb{N} \mapsto T_i(k) := 2kd_{\max} + t_0 + \sum_{j=1}^{i-1} d_j$;

1: $x_i(t) := l_i$ for $t_0 \le t \le T_i(0)$ and for $T_i(k) + 2d_i \le t \le T_i(k+1)$;
2: $x_i(t) := r_i$ for $t := T_i(k) + d_i$;

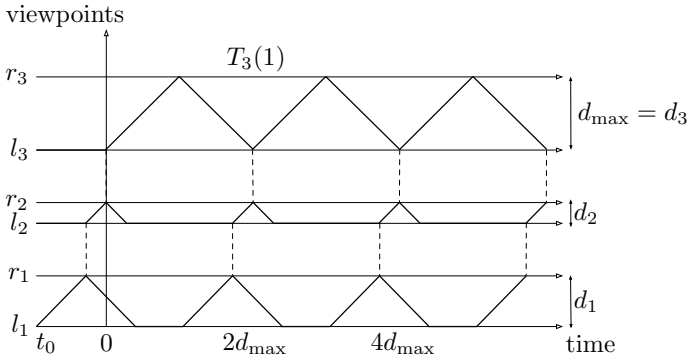


Fig. 4. A team trajectory with minimum up-latency generated with the procedure in Trajectory 3. Notice that each robot i communicates with the neighboring robot $i+1$ as soon as $x_i(t) = r_i$. A team trajectory with minimum down-latency is obtained by shifting in time the trajectory of robot 1, in a way that robot 2 communicates with robot 1 as soon as $x_2(t) = l_2$.

Remark 3 (Simplifications for clusters of equal length)**.** *Let $\{V_1, \dots, V_m\}$ be an optimal m-partition, and suppose that $d_{\max} = r_i - l_i$ for all $i \in \{1, \dots, m\}$. Then the up-latency and the down-latency can be made minimum and equal to $(m-2)d_{\max}$ by arranging the robots trajectories to be in opposite phase. Specifically, for $k = 0, 2, \dots,$ we set*

$$x_i(2kd_{\max}) = l_i, \quad x_i((2k+1)d_{\max}) = r_i,$$

if i is odd, and

$$x_i(2kd_{\max}) = r_i, \quad x_i((2k+1)d_{\max}) = l_i,$$

if i is even. Because of Lemma IV.1, the above trajectory has minimum latency. This particular case was studied in [2].

B. Lower bound for latency

We now consider the minimization of the latency criterion, and we restrict our attention to periodic team trajectories. To be more precise, let $\{V_1, \dots, V_m\}$ be an optimal m-partition of the environment, and let $d_{\max}$ denote the longest length of the clusters. We aim at finding a $2d_{\max}$-periodic team trajectory with image $\{V_1, \dots, V_m\}$ and minimum latency. Notice that, by imposing a periodicity of $2d_{\max}$, the refresh time of the trajectory, if finite, is also minimized.

We start by considering the pedagogical situation in which $d_i + d_{i+1} > d_{\max}$ for all $i \in \{1, \dots, m-1\}$. In the next Lemma, we show that the frequency of message exchange among the robots is limited by the periodicity of the trajectory.

Lemma IV.3 (Frequency of message exchange)**.** *Consider a $2d_{\max}$-periodic team trajectory, and let $d_i + d_{i+1} > d_{\max}$ for all $i \in \{1, \dots, m-1\}$. For any $t \in [0, T_f - 2d_{\max}]$ and for any $i \in \{2, \dots, m-2\}$, there*

[1]We focus on this family of trajectories, and we leave the more general optimization problem as the subject of future research. However, this family of trajectories is believed to contain an (unconstrained) optimal solution.

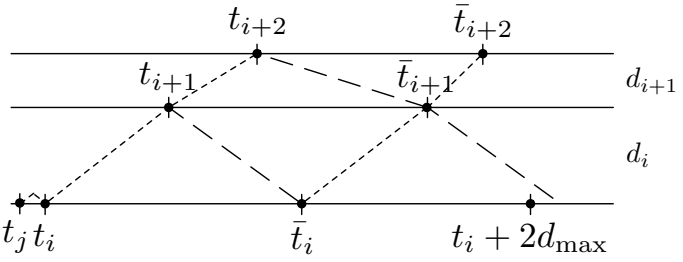


Fig. 5. As stated in Lemma IV.3, at most one communication sequence is possible within each period of length $2d_{\max}$. Here $d_i + d_{i+1} > d_{\max}$.

exist no two distinct sequences t_i, t_{i+1}, t_{i+2} *and* $\bar{t}_i, \bar{t}_{i+1}, \bar{t}_{i+2}$*, with* $t_j, \bar{t}_j \in \Phi_j$*,* $j = i, i+1, i+2$*, such that*

$$
\begin{aligned}
&t \le t_i \le t_{i+1} \le t_{i+2} \le t + 2d_{\max}\\
&t \le \bar{t}_i \le \bar{t}_{i+1} \le \bar{t}_{i+2} \le t + 2d_{\max}\\
&t_{i+1} \le \bar{t}_i\\
&t_{i+2} \le \bar{t}_{i+1}.
\end{aligned}
$$

Moreover, for any $i \in \{m, \dots, 4\}$*, there exist no two sequences* t_i, t_{i-1}, t_{i-2} *and* $\bar{t}_i, \bar{t}_{i-1}, \bar{t}_{i-2}$ *with* $t_j, \bar{t}_j \in \Phi_j$*,* $j = i, i-1, i-2$*, such that*

$$
\begin{aligned}
&t \le t_i \le t_{i-1} \le t_{i-2} \le t + 2d_{\max}\\
&t \le \bar{t}_i \le \bar{t}_{i-1} \le \bar{t}_{i-2} \le t + 2d_{\max}\\
&t_{i-1} \le \bar{t}_i\\
&t_{i-2} \le \bar{t}_{i-1}.
\end{aligned}
$$

Proof: Since $d_{i+1} + d_{i+2} > d_{\max}$, it follows $\max\{d_{i+1}, d_{i+2}\} > d_{\max}/2$. Let $d_{i+1} > d_{\max}/2$. By contradiction, if two distinct sequences t_i, t_{i+1}, t_{i+2} and $\bar{t}_i, \bar{t}_{i+1}, \bar{t}_{i+2}$ exist, with $t \le t_i \le t_{i+1} \le t_{i+2} \le t + 2d_{\max}$, $t \le \bar{t}_i \le \bar{t}_{i+1} \le \bar{t}_{i+2} \le t + 2d_{\max}$, $t_{i+1} \le \bar{t}_i$ and $t_{i+2} \le \bar{t}_{i+1}$, then the $(i+1)$-th robot travels its cluster four times. Since the speed of the robots is bounded by one, robot $i+1$ cannot travel its cluster four times in a period of $2d_{\max}$ (cf. Fig. 5). The second part of the theorem follows from an analogous reasoning. ■

Notice that in the above Lemma the index i belongs to the set $\{2, \dots, m-2\}$ (resp. $\{m, \dots, 4\}$) because we consider 3 consecutive communication instants, and because Φ_i denotes the sequence of times at which robots $i-1$ and i communicate. Because of Lemma IV.3, in a $2d_{\max}$-periodic team trajectory with $d_i + d_{i+1} > d_{\max}$ only one message can be passed from robot i to robot $i+3$ in a period of time of $2d_{\max}$. This limitation determines a lower bound on the latency of a periodic trajectory. Notice that eventual communication instants $t_j \in \Phi_i$, with $t \le t_j \le t_i \le t_{i+1}$, do not influence the latency, since all information can be transmitted at time t_i without affecting the latency performance.

Lemma IV.4 (Latency lower bound, simple case)**.** *Let* X *be a* $2d_{\max}$*-periodic team trajectory with* $d_i + d_{i+1} > d_{\max}$ *for all* $i \in \{1, \dots, m-1\}$*. Then*

$$\min_X \mathrm{LT}(X) \ge (m-2)d_{\max}.$$

Proof: Because of Lemma IV.3, a message can be transfered from robot i to robot $i+3$ in at most $2d_{\max}$ instants of time, by traveling the clusters V_{i+1} and V_{i+2}. Without losing generality, we let $d_{\max}$ be the time to pass a message form $i+1$ to $i+2$, and from $i+2$ to $i+3$. Notice that the same reasoning holds also for the time to transfer a message from $i+3$ to i. Therefore, the latency is lower bounded by $(m-2)d_{\max}$. ■

We now consider the situation in which an optimal m-partition does not verify the constraint $d_i + d_{i+1} > d_{\max}$ for all $i \in \{1, \dots, m-1\}$. Intuitively, for what concerns the latency, two consecutive clusters with

length d_i and d_{i+1} may be regarded as one single cluster of length $d_i + d_{i+1}$. Therefore, in order to use Lemma IV.3 and Lemma IV.4, we partition the clusters $\{V_1, \dots, V_m\}$ into groups such that the sum of the length of the clusters of two consecutive groups is greater than $d_{\max}$. In other words, let $\bar{V}_r = \{\bar{r}_1, \dots, \bar{r}_{\bar{m}}\}$ be the set of *right-extreme* viewpoints of the partition $\{V_1, \dots, V_m\}$ defined recursively as

$$\begin{aligned} \bar{r}_0 &:= l_1, \\ \bar{r}_i &:= \max_{j \in \{1,\dots,m\}} \{r_j \mid \sum_{k=p_i}^{j} d_k \leq d_{\max}\}, \quad i = 1, \dots, \bar{m}, \end{aligned}$$

where $p_i = \min\{1, \dots, m\}$ such that $l_{p_i} \geq \bar{r}_{i-1}$. Let $\bar{V}_l = \{\bar{l}_1, \dots, \bar{l}_{\bar{m}}\}$ be the set of *left-extreme* viewpoints defined recursively as

$$\begin{aligned} \bar{l}_1 &:= l_1, \\ \bar{l}_i &:= \min_{j \in \{1,\dots,m\}} \{l_j \mid l_j \geq \bar{r}_{i-1}\}, \quad i = 2, \dots, \bar{m}. \end{aligned}$$

Additionally, define the set of aggregated clusters $\bar{V} = \{\bar{V}_1, \dots, \bar{V}_{\bar{m}}\}$, where $\bar{V}_i$ contains all the clusters within $\bar{l}_i$ and $\bar{r}_i$, and let $\bar{d}_i$ be the sum of the length of the clusters in $\bar{V}_i$.

Lemma IV.5 (Latency lower bound, general case)**.** *Let X be a $2d_{\max}$-periodic team trajectory with image $\{V_1, \dots, V_m\}$, and let $\bar{m}$ be the number of aggregated clusters. Then,*

$$\min_X \mathrm{LT}(X) \geq (\bar{m} - 2)d_{\max} + (\bar{d}_1 - d_1) + (\bar{d}_{\bar{m}} - d_m).$$

Proof: Consider the clusters defined by the right extreme viewpoints, and notice that they verify $d_i + d_{i+1} > d_{\max}$. Then, the Theorem follows from Lemma IV.4, and from the fact that the minimum latency on the image $\{\bar{V}_1, \dots, \bar{V}_{\bar{m}}\}$ equals the minimum latency on the image $\{V_1, \dots, V_m\}$. The terms $\bar{d}_1 - d_1$ and $\bar{d}_{\bar{m}} - d_m$ are due to the fact that we are interested in delivering a message from robot 1 to robot m in the original configuration, and not on the aggregated chain. ■

C. Optimal team trajectory for latency

A team Trajectory with minimum refresh time and minimum latency is formally presented in Trajectory 4, where we specify the instants of time at which a robot changes its velocity, and we assume that it moves at maximum speed otherwise. An example is reported in Fig. 6, and here we give an informal description. Let $\{V_1, \dots, V_m\}$ be an optimal m-partition of a chain graph, and let $\{\bar{V}_1, \dots, \bar{V}_{\bar{m}}\}$ be the set of aggregated clusters. Recall that $\bar{V}_i$ is a subset of $\{V_1, \dots, V_m\}$, and that the sum of the length of two consecutive elements is larger than $d_{\max}$. The procedure in Trajectory 4 (lines $6-11$ and $18-23$) is such that the robots in the same group behave as a single robot assigned to the whole set of viewpoints. In other words, the motion of the robots in the same group is determined by a token passing mechanism, in which robot $i+1$ moves towards r_{i+1} only when $x_i(t) = r_i$ and $x_i(t^-) \neq r_i$, and, analogously, robot i moves towards l_i only when $x_i(t) = l_i$ and $x_i(t^-) \neq l_i$. Instead, lines $1-5$ and $12-17$ in Trajectory 4 guarantee the transfer of one message in a period of $2d_{\max}$ between three consecutive groups, and, consequently, the minimization of the latency. Indeed, since the sum of the length of two consecutive groups is larger than $d_{\max}$, because of Lemma IV.3, no more than one message can be transferred between three consecutive groups in a period of $2d_{\max}$.

Theorem IV.6 (Patrolling a chain graph at minimum refresh time and minimum latency)**.** *Let G be a chain graph with n viewpoints and let $m \leq n$ be the number of robots. Let $\mathcal{V}$ be an optimal m-partition of G computed by means of Algorithm 2 with tolerance ε. Let $d_{\max}$ be the dimension of $\mathcal{V}$, and let d_1 (resp. d_m) be the length of the first (resp. last) cluster in $\mathcal{V}$. Let $\bar{m}$ be the number of aggregated clusters, and let $\bar{d}_1$ (resp. $\bar{d}_{\bar{m}}$) be the length of the first (resp. last) aggregated cluster. A team trajectory with image*

Trajectory 4: *Minimum refresh time and latency trajectory (i-th robot)*

Input : $l_i := \min_{v \in V_i} v$, $r_i := \max_{v \in V_i} v$, $d_{\max} := \max_j r_j - l_j$, $R :=$ set containing the identifiers of the robots in the same group as i, $k :=$ index of the aggregated cluster $\{1, \dots, \bar{m}\}$;
Require: optimal partition of the chain graph;

case l_i *is a left-extreme*
  if k *is odd* **then**
    $x_i(t) = l_i$ with $t = 0, 2d_{\max}, 4d_{\max}, \dots$;
  else if k *is even* **then**
    $x_i(t) = l_i$ with $t = d_{\max}, 3d_{\max}, 5d_{\max}, \dots$;
case l_i *is not a left-extreme*
  $\delta_i := \sum_{j \in R, j < i} d_j$;
  if k *is odd* **then**
    $x_i(t) = l_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = 0, 2d_{\max}, 4d_{\max}, \dots$;
  else if k *is even* **then**
    $x_i(t) = l_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = d_{\max}, 3d_{\max}, 5d_{\max}, \dots$;
case r_i *is a right-extreme*
  $\delta_i := d_{\max} - \sum_{j \in R} d_j$
  if k *is odd* **then**
    $x_i(t) = r_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = d_{\max}, 3d_{\max}, 5d_{\max}, \dots$;
  else if k *is even* **then**
    $x_i(t) = r_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = 0, 2d_{\max}, 4d_{\max}, \dots$;
case r_i *is not a right-extreme*
  $\delta_i := d_{\max} - \sum_{j \in R, j \le i} d_j$;
  if k *is odd* **then**
    $x_i(t) = r_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = d_{\max}, 3d_{\max}, 5d_{\max}, \dots$;
  else if k *is even* **then**
    $x_i(t) = r_i$ for $\tau - \delta_i \le t \le \tau + \delta_i$, with $\tau = 0, 2d_{\max}, 4d_{\max}, \dots$;

$\mathcal{V}$, minimum refresh time $2d_{\max}$, and minimum latency $(\bar{m}-2)d_{\max} + \bar{d}_1 - d_1 + \bar{d}_{\bar{m}} - d_m$ is computed as in Trajectory 4. Moreover, the time complexity of designing such trajectory is $O(n \log(\varepsilon^{-1}))$.

Proof: By inspection, the team Trajectory described in Trajectory 4 is $2d_{\max}$-periodic, and therefore it has minimum refresh time. Moreover, by construction, the communications at the extreme viewpoints happen every $2d_{\max}$ instants of time, so that the latency is equal to $(\bar{m}-2)d_{\max} + \bar{d}_1 - d_1 + \bar{d}_{\bar{m}} - d_m$, and hence, by Lemma IV.5, minimum. Regarding the computational complexity, notice that it is determined by the computation of the optimal m-partition $\mathcal{V}$, and hence, by Lemma III.5, it equals $O(n \log(\varepsilon^{-1}))$. ■

V. Distributed synchronization algorithm on a chain roadmap

In the previous sections we have shown that, for the computation of a minimum refresh time and latency team trajectory, first an optimal m-partition of the roadmap needs to be found, and, then, a synchronization of the motion of the robots needs to be achieved to ensure communication between neighboring robots. The distributed computation of an optimal m-partition follows directly from Algorithm 2, by letting the robots compute the left-induced partition of length l in a distributed way. A simple solution consists of the following three operation:

(i) the robots gather at the leftmost viewpoint, and
(ii) determine the cardinality of the team and elect a leader;
(iii) the leader computes an optimal left-induced partition, and assigns a different cluster to each robot.

Notice that, by only assuming the capability of detecting the viewpoints in the roadmap (in particular the leftmost and rightmost viewpoint) the robots can distributively compute an optimal partition. Indeed, the leader can simply travel the roadmap, and evaluate if, for a given length ρ, the cardinality of the

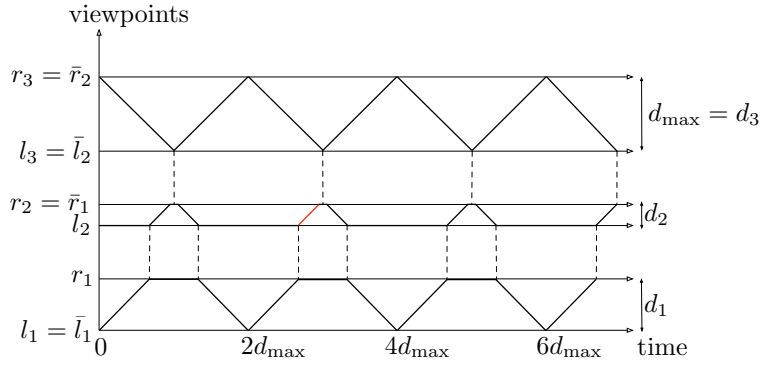


Fig. 6. A team trajectory with minimum refresh time ($2d_{\max}$) and minimum latency ($(\bar{m}-2)d_{\max} + d_1 + d_2 - d_1 + d_3 - d_3 = d_2$) is here reported. A message with delivery time equal to d_2 is reported in red. The right-extreme viewpoints are $\{\bar{r}_1, \bar{r}_2\}$, the left-extreme viewpoints are $\{\bar{l}_1, \bar{l}_2\}$, and $\bar{m} = 2$. The represented team trajectory has minimum refresh time ($2d_{\max}$) and minimum latency ($(\bar{m}-2)d_{\max} + (d_1 + d_2) - d_1 + d_3 - d_3 = d_2$). A message with delivery time equal to d_2 is reported in red.

corresponding left-induced partition is greater, equal, or smaller than the cardinality of the team. We believe that, by simple programming operations, the above procedure can be improved, so as to handle reconfigurations due to addition or removal of robots in the team. In this section we focus instead on the design of a distributed feedback law to synchronize the motion of the robots so as to guarantee the minimization of the latency of the trajectory.

Recall that $x_i(t)$ denotes the position on the chain of the robot i at time t. Moreover, let the i-th cluster of an optimal m-partition be delimited by l_i and r_i. Let $\text{dir}_i \in \{-1, 0, 1\}$ denote the direction of motion of robot i. Let *c-time* denote the current time of simulation, let *a-time* be the time at which a robot arrives at his right extreme. Let *n-meet(i,j)* be a function that returns the number of times that the robots i and j have communicated. Let *Timer(δ)* be a function that returns 1 after a time interval of length δ. An algorithm for the robots to distributively converge to a minimum refresh time and latency team trajectory is in Algorithm 5. It should be noticed that the algorithm assumes the knowledge of an optimal partitioning of the chain graph, and of the left- and right-extreme sets.

Algorithm 5 is informally described as follows. First, the velocity of a robot changes only when its position coincides with the leftmost or the rightmost of the assigned viewpoints. When a robot reaches an extreme viewpoint, it waits until a communication with the neighboring robot happens (lines $7-8$). This mechanism determines the feedback behavior of our procedure. The behavior of a robot after a communication is determined by the lines $9-21$, which reproduce the optimal behavior described in Trajectory 4. To be more precise, the function *Token(i, j)* coordinates the motion of the robots in the same group (see Section IV-C), so that they move as if there was a single robot sweeping the viewpoints in the same group. The function *Timer(δ_i)*, instead, ensures the maximum frequency of messages exchange between consecutive groups, so as to minimize the latency of the resulting trajectory.

Theorem V.1 (Optimal team trajectory)**.** *Let X be the team trajectory generated by Algorithm 5. There exists a finite time after which X has minimum refresh time and latency.*

Proof: Let $\{V_1, \ldots, V_m\}$ be an optimal m-partition, let $\bar{V}_r = \{\bar{r}_1, \ldots, \bar{r}_{C^*}\}$ be the set of right-extreme viewpoints, and let $\bar{V}_l = \{\bar{l}_1, \ldots, \bar{l}_{C^*}\}$ be the set of left-extreme viewpoints. Let R denote the set of robots patrolling a viewpoint between $\bar{l}_i$ and $\bar{r}_i$, where $\bar{l}_i$ and $\bar{r}_i$ are as previously defined. First, notice that robot 1 (resp. m) sets $\dot{x}_1(t) = 1$ (resp. $\dot{x}_m(t) = -1$) as soon as $x_1(t) = l_1$ (resp. $x_m(t) = r_m$). Second, the function *Token*(i, j) guarantees that, when $i, j \in R$ communicate, exactly one robot among i, j maintains a zero velocity and in an alternate way. Therefore, after a finite time T_i, independent of the initial robots motion direction, the velocities of the robots in R are such that, upon communication, $\dot{x}_i(t) = \dot{x}_{i-1}(t^-)$ and $\dot{x}_{i-1}(t) = 0$. In other words, after T_i, the robots in R behave as a single robot sweeping the segments between $\bar{l}_i$ and $\bar{r}_i$. Finally, the function *Timer*(τ) and the parameter τ guarantee that the communications

Algorithm 5: *Minimum refresh time and latency team trajectory (i-th robot)*

Input : $l_i := \min_{v \in V_i} v$, $r_i := \max_{v \in V_i} v$, $d_{\max} := \max_j r_j - l_j$, $R :=$ set containing the identifiers of the robots in the same group as i;
Set : $\delta_i := (d_{\max} - \sum_{j \in R} d_j)/2$, $\text{dir}_i \in \{1, -1\}$;
Require : optimal partition of the chain graph; $l_i \leq x_i(0) \leq r_i$;

while *true* **do**
  case $i = 1$ *and* $x_1(t) = l_1$
    $\text{dir}_1 := 1$;
  case $i = m$ *and* $x_m(t) = r_m$
    $\text{dir}_m := -1$;
  case *($x_i(t) = l_i$ and $x_{i-1}(t) \neq r_{i-1}$) or ($x_i(t) = r_i$ and $x_{i+1}(t) \neq l_{i+1}$)*
    $\text{dir}_i := 0$;
  case $x_i(t) = l_i$ *and* $x_{i-1}(t) = r_{i-1}$
    if $l_i \in \bar{V}_l$ **then**
      receive τ from robot $i - 1$;
      $\text{dir}_i := \text{Timer}(\tau)$;
    else if $l_i \notin \bar{V}_l$ **then**
      $\text{dir}_i := \text{Token}(i, i - 1)$;
  case $x_i(t) = r_i$ *and* $x_{i+1}(t) = l_{i+1}$
    $\tau := \max\{0, \textit{a-time} + \delta_i - \textit{c-time}\}$;
    send τ to robot $i + 1$;
    if $r_i \in \bar{V}_r$ **then**
      $\text{dir}_i := -\text{Timer}(\delta_i + \tau)$;
    else if $r_i \notin \bar{V}_r$ **then**
      $\text{dir}_i := \text{Token}(i, i + 1)$;
  $\dot{x}_i(t) := \text{dir}_i$;

Function Token(i,j)

case $i > j$ **if** *n-meet(i,j) is even* **then return** *1*
case $i < j$ **if** *n-meet(i,j) is odd* **then return** *-1*
otherwise return *0*

at the extreme viewpoints happen every $2d_{\max}$ instants of time. We conclude that the trajectory generated by Algorithm 5 converges in finite time to a team trajectory with minimum refresh time and latency. ■

It should be observed that, differently from the team trajectories presented in the previous sections, Algorithm 5 contains a feedback procedure to synchronize the motion of the robots. Our algorithm is independent of the initial configuration, and, as it is shown in the next section, it is robust to a certain class of robot failures.

VI. A CASE STUDY

Given the theoretical nature of this work, in this section we propose a simulation study to show the effectiveness of our procedures. For our simulations, we use the Matlab® simulation environment, and we model the robots as holonomic vehicles of zero dimension. The communication edges and the motion paths are described by a given roadmap.

Consider the chain roadmap with 30 viewpoints in Fig. 7. Suppose that 10 robots are assigned to the patrolling task. In order to obtain a team trajectory with minimum refresh time, an optimal 10-partition of the roadmap is computed (cf. Fig. 7). Additionally, to obtain a minimum latency team trajectory, each robot is assigned to a different cluster, its initial position is chosen randomly inside its cluster, and its velocity is initialized randomly. The motion of each robot is then determined by Algorithm 5. The resulting

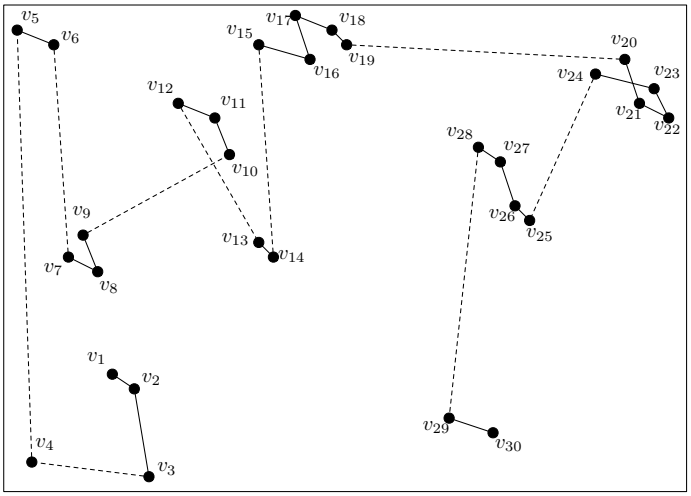


Fig. 7. The figure shows a robotic roadmap with sensor coverage and communication connectivity of a two floors building. Crossing edges corresponds to corridors at different floors. A 10-partition of the roadmap is here reported. The dashed edges are not traveled by any robot.

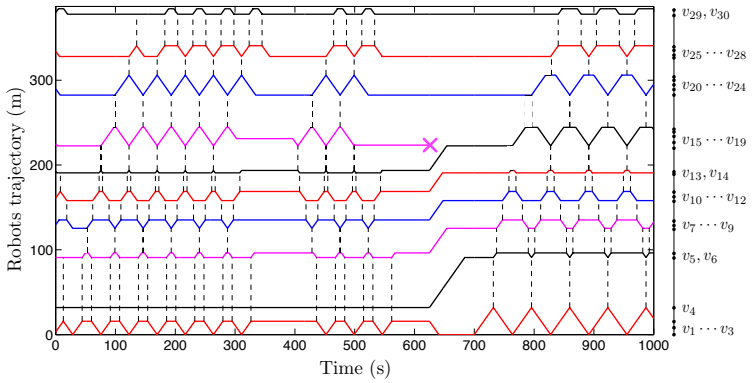


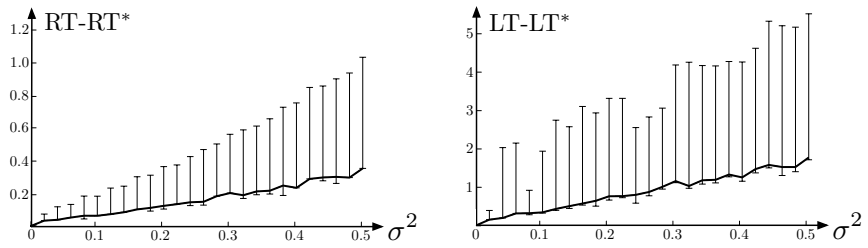
Fig. 8. For the roadmap of Fig. 7, the team trajectory obtained with Algorithm 5 is reported here. The dashed lines denote a communication among two neighboring robots. At time $t = 200$ the robots have synchronized their motion, and from that moment up to time $t = 300$ the team trajectory has minimum refresh time and latency. From time $t = 300$ up to time $t = 400$, robot 7 undergoes a temporary failure, causing all the other robots to lose synchronization. The team of robots synchronizes again when robot 7 resume its motion. At time $t = 620$, the failure of robot 7 is detected by the remaining robots, which compute and synchronize on a new partition.

team trajectory is in Fig. 8, where the team of robots synchronize on a minimum refresh time and latency team trajectory after a finite transient.

We now test the robustness of our synchronization procedure. As first case study, we consider a temporary stopping failure, in which a robot stops for a certain time interval. For instance, suppose that robot 7 stops from time 300 up to time 400 (cf. Fig. 8). Notice that, after the failure, each robot j, with $j < 7$, gathers at r_j, and each robot k, with $k > 7$, gathers at l_k waiting for a communication with the corresponding neighboring robot. As soon as robot 7 resumes its functionalities, the team of robots recover the desired synchronization. Notice that the transient failure of robot 7 can be easily detected by its neighbors by means of a timer mechanism with a predefined threshold.

As a second case study, we let the robots actuation be affected by noise, so that the speed of the robots becomes a random variable with a certain distribution.[2] Precisely, let $\dot{x}_i = \text{dir}_i + w_i$ be the equation describing the dynamics of robot i, where w_i is a zero mean Gaussian variable with variance σ^2 $[(\text{m/s})^2]$. We let $\sigma^2 \in \{0, 0.02, \ldots, 0.5\}$ and we run 100 simulations for each possible value of σ^2 on the roadmap of Fig. 7. The refresh time and the latency of the team trajectories obtained with Algorithm 5 are plotted in

[2]The case in which a robot fails at seeing a neighboring robot for a certain interval of time can be modeled analogously.



(a) (b)

Fig. 9. For the roadmap in Fig. 7, 100 team trajectories have been generated with Algorithm 5 for each value of σ^2. The average refresh time and the average latency are plotted as continuous lines in Fig. 9(a) and Fig. 9(b) respectively. The bars indicate the minimum and the maximum value of the refresh time and the latency, respectively. Notice that the average team trajectory performance degrade gracefully with the noise variance.

Fig. 9(a) and in Fig. 9(b), respectively, as a function of σ^2. Note that the performance degrade gracefully with the noise magnitude.

As third and final case study, we consider the situation in which a robot definitively stops.[3] The remaining robots need to compute a new optimal partition and to synchronize in order to guarantee an optimal patrolling of the environment. Notice that for the computation of such a partition by Algorithm 2 the chain graph and the number of the robots is required. Suppose that the failure of the robot 7 is detected at time 620 by the well-behaving robots, and assume that each robot knows the chain roadmap and the number of operative robots. Algorithm 2 and Algorithm 5 allow the team to synchronize on a new team trajectory with minimum refresh time and latency. Notice that the initial and the final partitions do not coincide.

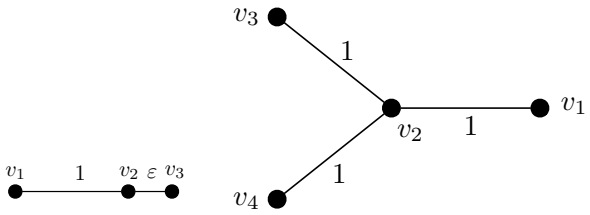
VII. Approximation algorithms and heuristics for general roadmaps

The problem of designing minimum refresh time and latency team trajectories on a chain roadmap has been discussed. In this section we consider the more general cases of tree and cyclic roadmap, we characterize the computational complexity of determining optimal trajectories, and we describe two approximation methods with performance guarantees. The results we are going to present are intended for a team of more than one robot. Indeed, if only one robot is assigned to the patrolling task, then a minimum refresh time trajectory follows from the computation of the shortest tour through the viewpoints, for which efficient approximation algorithms already exist [28].

A. Minimum refresh time team trajectory on a tree roadmap

Let $T = (V, E)$ denote an undirected, connected, and acyclic roadmap (tree). Recall that a vertex path is a sequence of vertices such that any pair of consecutive vertices in the sequence are adjacent. A tour is a vertex path in which the start and end vertices coincide, and in which every vertex of T appears at least once in the sequence. A depth-first tour of T is a tour that visits the vertices V in a depth-first order [29]. Let DFT(T) denote the length of a depth first tour of T. Notice that the length of a depth-first tour of a connected tree equals twice the sum of the length of the edges of the tree, and that any depth-first tour is a shortest tour visiting all the vertices. We now show that, for the case of tree roadmap, the set of cyclic-based and partition-based trajectories described in [15] does not contain, in general, a minimum refresh time trajectory. Recall that in a cyclic-based strategy the robots travel at maximum speed and equally spaced along a minimum length tour visiting all the viewpoints. Consider the tree roadmap of Fig. 10(a), and suppose that two robots are assigned to the patrolling task. Clearly, the minimum refresh time is 2ε, while the refresh time of a cyclic strategy equals $1 + \varepsilon$. Consider now the tree roadmap in Fig. 10(b), where the edges have unit length, and assume that two robots are in charge of the patrolling task.

[3]The case of additional robots joining the team is handled in a similar way.



(a) (b)

Fig. 10. Two examples of tree roadmap. For the roadmap in Fig. 10(a) the family of cyclic based trajectories does not contain a minimum refresh time team trajectory. Instead, for the roadmap in Fig. 10(b) the family of partitioned based trajectories does not contain a minimum refresh time team trajectory.

Observe that any partition of cardinality 2 contains a chain of length 2, so that, since only one robot is assigned to each cluster, the minimum refresh time that can be obtained is 4. Suppose, instead, that the robots visit the vertices of the roadmap as specified in Table I, where $x(t)$ denotes the position of a robot at time t. Since the refresh time of the proposed trajectory is 3, we conclude that neither the cyclic-based nor the partition-based strategy may lead to a minimum refresh time team trajectory on a tree roadmap.

TABLE I

Robot	$x(0)$	$x(1)$	$x(2)$	$x(3)$	$x(4)$	$x(5)$	$x(6)$	$x(7)$
1	v_1	v_2	v_4	v_2	v_3	v_2	v_1	$\cdots$
2	v_2	v_3	v_2	v_1	v_2	v_4	v_2	$\cdots$

We now introduce some definitions. Let X be a team trajectory on the tree roadmap T. We say that the edge $(v_j, v_z) \in E$ is *used* by X if there exists $i \in \{1, \dots, m\}$ and $(t_1, t_2) \in [0, \text{RT}(X)]^2$ such that $x_i(t_1) = v_j$ and $x_i(t_2) = v_z$, and it is *unused* otherwise. Note that, because in a tree there exists only one path connecting two vertices, the above condition ensures that the edge (j, z) is traveled by the robot i. Let $\bar{E}$ denote the set of unused edges, and let F_T be the forest obtained from T by removing the edges $\bar{E}$ from E, i.e., the collection of vertex-disjoint subtrees $\{T_1, \dots, T_k\}$, with $T_i = (V_i, E_i)$, such that $V = \cup_{i=1}^{k} V_i$ and $E_i \subseteq E$, for each $i \in \{1, \dots, k\}$. Let m_i be the number of robots that visit at least one vertex of T_i in the interval $[0, \text{RT(X)}]$, and note that $m_i > 0$ in a finite refresh time trajectory. Let $M = \{m_1, \dots, m_k\}$. Notice that the same subtree collection can be associated with different team trajectories. We say that a team trajectory is *efficient* if its refresh time is the smallest among all the team trajectories associated with the same subtree collection.

Theorem VII.1 (Efficient team trajectory)**.** *Let (F_T, M) be the subtree collection associated with the team trajectory X on the tree roadmap T, where $F_T = \{T_1, \dots, T_k\}$, and $M = \{m_1, \dots, m_k\}$. Then, X is efficient if*

$$\text{RT}(X) = \max_{j \in \{1, \dots, k\}} \text{DFT}(T_j)/m_j.$$

Proof: Let $i \in \{1, \dots, k\}$, and let m_i be the number of robots assigned to T_i. Notice that the robots in T_i travel, in total, at least $\text{DFT}(T_i)$ to visit all the vertices. Since the speed of the robots is bounded by 1, the smallest refresh time for the vertices of T_i is $\text{DFT}(T_i)/m_i$. ■

An efficient team trajectory, can be computed with the following procedure. See Table I for an example.

Lemma VII.2 (Efficient team trajectory computation)**.** *Let (F_T, M) be a subtree collection of a tree roadmap, where $F_T = \{T_1, \dots, T_k\}$, and $M = \{m_1, \dots, m_k\}$. An efficient team trajectory is as follows: for each $i \in \{1, \dots, k\}$,*

(i) *compute a depth-first tour τ_i of T_i,*
(ii) *equally space m_i robots along τ_i, and*

(iii) *move the robots clockwise at maximum speed on τ_i.*

Proof: Since every vertex of $T_i \in F_T$ appears at least once in a depth first tour τ_i of T_i, and the robots move with maximum speed and equally spaced along τ_i, every vertex is visited at most every $\mathrm{DFT}(T_i)/m_i$. The statement follows. ■

Let $P(m)$ be the partition set of m, i.e., the set of all the sequences of integers whose sum is m. The following problem is useful to characterize the complexity of designing minimum refresh time trajectories on a tree roadmap.

Problem 2 (Optimal subtree collection)**.** *Let T be a tree roadmap and let m be the number of robots. Find a subtree collection (F_T, M) that minimizes $\max_{j\in\{1,\dots,|F_T|\}} \mathrm{DFT}(T_j)/m_j$ subject to $M \in P(m)$ and $|F_T| = |M|$.*

Lemma VII.3 (Equivalent problem)**.** *For the case of a tree roadmap, the Team refresh time problem and the Optimal subtree collection problem are equivalent.*

Proof: As a consequence of Theorem VII.1, the minimum refresh time on a tree roadmap T can be written as $\min_{(F_T,M)} \max_{j\in\{1,\dots,k\}} \mathrm{DFT}(T_j)/m_j$, where (F, M) is a subtree collection of T, and $|M| = |F_T| = k \leq m$. It follows that a solution to Problem 1 can be derived in polynomial time from a solution to Problem 2 by using the procedure described in Lemma VII.2. Suppose now we have a solution to Problem 1. Then an optimal subtree collection follows from the identification of the unused edges. We conclude that the two optimization problems are equivalent. ■

We now state our main result on the design of minimum refresh time team trajectory on a tree roadmap.

Theorem VII.4 (Computing a minimum refresh time team trajectory on a tree)**.** *Let T be a tree roadmap with n vertices, and let m be the number of robots. A minimum refresh time team trajectory on T can be computed in $O((m-1)!n)$ time.*

Proof: Recall from [30] that an optimal subtree collection can be computed in $O((m-1)!n)$. Then, by using Lemma VII.3 and Lemma VII.2, the claimed statement follows. ■

As a consequence of Theorem VII.4, the problem of designing minimum refresh time team trajectories on a tree roadmap is computationally *easy* for any finite number of robots. In our design procedure, we first compute an optimal subtree collection of the given tree, and then we schedule the robots trajectory according to Lemma VII.2.

B. Minimum refresh time team trajectory on a cyclic roadmap

In this section we propose two approximation methods for the Team refresh time problem in the case of a cyclic, i.e., not acyclic, roadmap. These solutions are obtained from a transformation of the cyclic roadmap into an acyclic roadmap.

Let $G = (V, E)$, with $|V| = n$, be an undirected and connected roadmap. Note that there exists an open tour τ with at most $2n-4$ edges that visits all the vertices.[4] We construct a chain roadmap Γ from τ by doubling its repeated vertices and edges, so that Γ has at most $2n-3$ vertices and at most $2n-4$ edges, and such that the length of the i-th edge of Γ equals the length of the i-th edge of τ (cf. Fig. 11). Our first approximation method consists of applying Algorithm 5 to an optimal m-partition of Γ.

Theorem VII.5 (Performance ratio)**.** *Let G be a connected roadmap, let n be the number of vertices of G, and let γ be ratio of the longest to the shortest length of the edges of G. Let RT^* be the minimum refresh time on G. Let τ be an open tour with $2n-4$ edges that visits all the n vertices, and let Γ be the chain roadmap associated with τ. Let RT^*_Γ be the minimum refresh time on Γ. Then*

$$RT^*_\Gamma \leq \left(\frac{n-2}{n}\right) 8\gamma RT^*.$$

[4]An open tour with at most $2n-4$ edges that visits all the vertices can be constructed starting from a leaf of a spanning tree of G.

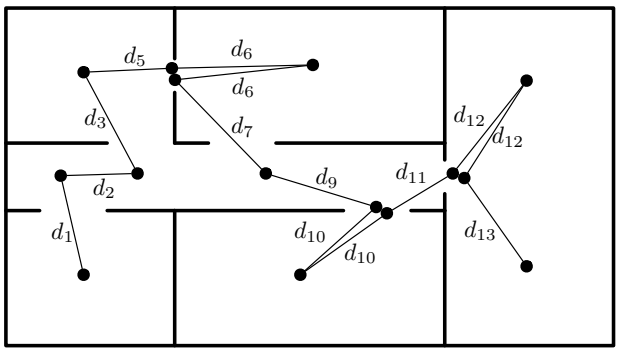


Fig. 11. The figure shows a chain roadmap associated with the cyclic roadmap of Fig. 1. Notice that the cycles in Fig. 1 have been broken. Moreover, 3 vertices (3 edges) of Fig. 1 are repeated twice in the chain.

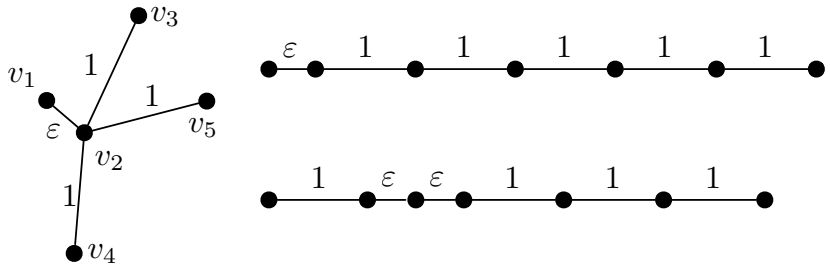


Fig. 12. A tree roadmap and two corresponding chain roadmaps. If the number of robots is 4, then the performance ratio $\mathrm{RT}^*_\Gamma/\mathrm{RT}^*$ grows with ε^{-1}.

Proof: Let $\underline{w}$ be the shortest length of the edges of G, and note that the length of Γ is upper bounded by $2(n-2)\gamma\underline{w}$. It follows that $\mathrm{RT}^*_\Gamma \leq \frac{4(n-2)\gamma\underline{w}}{m}$. Since $m < n$ by assumption, some robots need to move along G for all the viewpoints to be visited. Because each robot can visit only a vertex at a time, at least $\left\lceil \frac{n}{m} - 1 \right\rceil$ steps are needed to visit all the vertices of G, and therefore $\mathrm{RT}^* \geq \left\lceil \frac{n}{m} - 1 \right\rceil \underline{w} \geq \frac{1}{2}\frac{n}{m}\underline{w}$. By taking the ratio of the two quantities the statement follows. ■

It should be noticed that, when γ grows, the performance of our procedure might degrade. For instance, suppose that the roadmap is as in Fig. 12, and suppose that 4 robots are assigned to the patrolling task. As long as $\varepsilon < 1$, a minimum refresh time strategy requires one robot to patrol the viewpoints $\{v_1, v_2\}$, while the second, third, and fourth robot stay on the viewpoints v_3, v_4, and v_5 respectively. It follows that $\mathrm{RT}^* = 2\varepsilon$. On the other hand, an optimal m-partition of any chain graph associated with a tour that visits all the viewpoints has dimension at least 1. Consequently, the refresh time of the team trajectory obtained with Algorithm 5 equals 2, and the ratio $\mathrm{RT}^*_\Gamma/\mathrm{RT}^*$ grows proportionally to ε^{-1}.

We next describe a polynomial time constant factor approximation algorithm for the minimum refresh time problem. Given a roadmap $G = (V, E)$ and a positive integer $k < |V|$, we define a path cover of cardinality k as the collection of paths $\{p_1, \ldots, p_k\}$ such that $V \subseteq \bigcup_{i=1}^{k} p_i$. Let the cost of a path equal the sum of the lengths of its edges. The min-max path cover problem asks for a minimum cost path cover for the input graph, where the cost of a cover equals the maximum cost of a path in the cover. The following result is known.

Theorem VII.6 (Min-max path cover [7])**.** *There exists a* 4*-approximation polynomial algorithm for the NP-hard min-max path cover problem.*

Following Theorem VII.6, given a graph G, there exists a polynomial time algorithm that computes a path cover of G with cost at most 4 times greater than the cost of any path cover of G. We now state our approximation result for the *NP-hard* Team refresh time problem.

Lemma VII.7 (8-approximation refresh time)**.** *There exists an* 8*-approximation polynomial algorithm for*

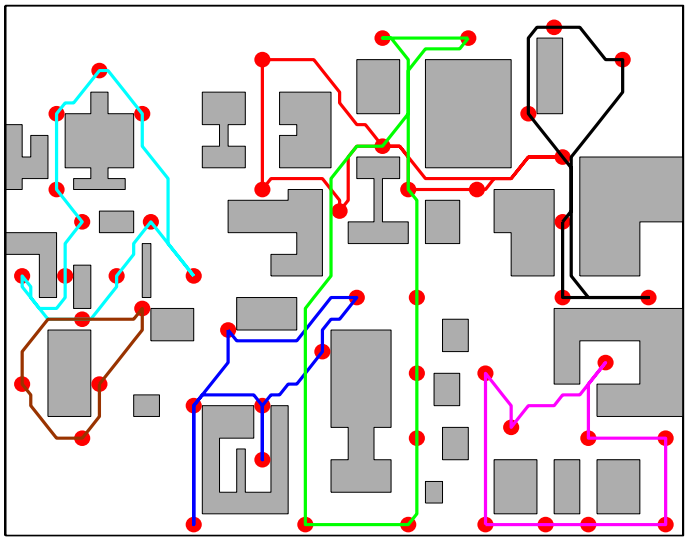


Fig. 13. The picture shows the trajectories of 7 robots for the patrolling of a part of the UCSB campus. The viewpoints (red circles) are chosen so as to provide sensor coverage of the whole area. For the design of the patrolling trajectory, a roadmap is first constructed as described in Section II. Then, a path covering of cardinality 7 is computed with the procedure in [7], and each robot is assigned a different path. Finally, the trajectory of each robot consists in sweeping at maximum speed the tour obtained by shortcutting the assigned path. The refresh time of the proposed team trajectory is proven to be within a factor of 8 of the minimum refresh time for the given roadmap.

the NP-hard Team refresh time problem.

Proof: Let $\{p_1, \ldots, p_m\}$ be a 4-approximation path cover of the graph G. Note that the length of each path is within 4RT^*. Indeed, in a minimum refresh time team trajectory starting at time 0 and with unitary velocity, every vertex is visited within time RT^*. Let X be a team trajectory obtained by letting robot i sweep at maximum speed the path p_i. Clearly, $\text{RT}(X) \leq 8\text{RT}^*$. Because of Theorem VII.6, the team trajectory X can be computed in polynomial time. ■

Following Lemma VII.7, for any given roadmap and any number of robots, a team trajectory with refresh time within a factor of 8 of the optimal refresh time can be constructed by computing a path covering of the roadmap, and by assigning a different path to each robot. An example is in Fig. 13, a movie of which is included in the multimedia material.

Remark 4 (Improving the team trajectory)**.** *Several existing heuristics can be used to improve upon the trajectories in Fig. 13. For instance, since the robots move in a metric space, shortcutting techniques may be applied [31]. Because these heuristics do not guarantee an improvement of the optimality gap of our trajectories, they are not considered in this work, and they are left as the subject of future investigation.*

VIII. Conclusion and Future Work

The design of team trajectories to cooperatively patrol an environment has been discussed. After defining the problem and the performance criteria, we analyze the computational complexity of the design problem as a function of the shape of the environment to be patrolled. For the case of a chain environment, we describe a polynomial algorithm to compute minimum refresh time and latency team trajectories. For the case of a tree environment, under the technical assumption of a constant number of robots, we identify a polynomial time algorithm to compute minimum refresh time team trajectories. Finally, the general case of cyclic environment is shown to be *NP-hard*, and two approximation algorithms with performance guarantees have been proposed.

Interesting aspects requiring further investigation include a throughout study of the latency optimization problem for cyclic roadmaps, the development of more efficient approximation algorithms, and the introduction of a more general communication framework, in which the robots are allowed to communicate while traveling the edges of the roadmap. The study of average performance criteria and the extension

to dynamically changing environments are also of interest. Finally, an hardware implementation of our algorithms would further strengthen our theoretical findings.

REFERENCES

[1] J. Clark and R. Fierro, "Mobile robotic sensors for perimeter detection and tracking," *ISA Transactions*, vol. 46, no. 1, pp. 3–13, 2007.

[2] D. B. Kingston, R. W. Beard, and R. S. Holt, "Decentralized perimeter surveillance using a team of UAVs," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1394–1404, 2008.

[3] S. Susca, S. Martínez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 2, pp. 288–296, 2008.

[4] Y. Elmaliach, A. Shiloni, and G. A. Kaminka, "A realistic model of frequency-based multi-robot polyline patrolling," in *International Conference on Autonomous Agents*, Estoril, Portugal, May 2008, pp. 63–70.

[5] B. C. Tansel, R. L. Francis, and T. J. Lowe, "Location on networks: a survey. Part I: the p-center and p-median problems," *Management Science*, vol. 29, no. 4, pp. 482–497, 1983.

[6] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[7] E. M. Arkin, R. Hassin, and A. Levin, "Approximations for minimum and min-max vehicle routing problems," *Journal of Algorithms*, vol. 59, no. 1, pp. 1–18, 2006.

[8] M. Batalin and G. S. Sukhatme, "The design and analysis of an efficient local algorithm for coverage and exploration based on sensor network deployment," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 661–675, 2007.

[9] N. Agmon, S. Kraus, and G. A. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *IEEE Int. Conf. on Robotics and Automation*, Pasadena, CA, May 2008, pp. 2339–2345.

[10] F. Amigoni, N. Basilico, and N. Gatti, "Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments," in *IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, May 2009, pp. 2005–2010.

[11] A. Marino, L. Parker, G. Antonelli, and F. Caccavale, "Behavioral control for multi-robot perimeter patrol: A finite state automata approach," in *IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, May 2009, pp. 3350–3355.

[12] D. A. Anisi, P. Ögren, and X. Hu, "Cooperative minimum time surveillance with multiple ground vehicles," *IEEE Transactions on Automatic Control*, no. 99, 2010, to appear.

[13] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre, "Recent advances on multi-agent patrolling," in *Advances in Artificial Intelligence - SBIA 2004*. Springer, 2004, vol. 3171, pp. 474–483.

[14] A. Machado, G. Ramalho, J. D. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," in *Multi-Agent-Based Simulation II*, ser. Lecture Notes in Computer Science. Springer, 2003, pp. 155–170.

[15] Y. Chevaleyre, "Theoretical analysis of the multi-agent patrolling problem," in *IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, Beijing, China, Sep. 2004, pp. 302–308.

[16] S. L. Smith and D. Rus, "Multi-robot monitoring in dynamic environments with guaranteed currency of observations," in *IEEE Conf. on Decision and Control*, Atlanta, GA, USA, Dec. 2010, pp. 514–521.

[17] R. E. Korf, "Real-time heuristic search: New results," in *National Conf. on Artificial Intelligence*, San Paul, MN, Aug. 1988, pp. 139–144.

[18] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859–865, 1991.

[19] M. A. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," *Information and Computation*, vol. 176, no. 1, pp. 1–21, 2002.

[20] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Efficiently searching a graph by a smell-oriented vertex process," *Annals of Mathematics and Artificial Intelligence*, vol. 24, no. 1, pp. 211–223, 1998.

[21] F. Pasqualetti, A. Franchi, and F. Bullo, "On optimal cooperative patrolling," in *IEEE Conf. on Decision and Control*, Atlanta, GA, USA, Dec. 2010, pp. 7153–7158.

[22] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006, available at `http://planning.cs.uiuc.edu`.

[23] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[24] K. J. Obermeyer, A. Ganguli, and F. Bullo, "Multi-agent deployment for visibility coverage in polygonal environments with holes," *International Journal on Robust and Nonlinear Control*, Sep. 2010, to appear.

[25] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "MAC vs. PC: Determinism and randomness as complementary approaches to robotic exploration of continuous unknown domains," *International Journal of Robotics Research*, vol. 19, no. 1, pp. 12–31, 2000.

[26] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli, "The sensor-based random graph method for cooperative robot exploration," *IEEE/ASME Transactions on Mechatronics*, vol. 14, no. 2, pp. 163–175, 2009.

[27] M. R. Garey and D. S. Johnson, *Computers and Intractability*. Springer, 1979.

[28] S. Arora, "Polynomial-time approximation schemes for the Euclidean TSP and other geometric problems," *Journal of the ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[29] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*, ser. Monographs on Discrete Mathematics and Applications. SIAM, 2000.

[30] H. Nagamochi and K. Okada, "A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree," *Discrete Applied Mathematics*, vol. 140, no. 1-3, pp. 103–114, 2004.

[31] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.